REMARKS/ARGUMENTS

Claims 1-4, 6-14, 16-18, 20-24, 26-28, and 30 are pending in the present application. Claims 5, 15, 19, 25, and 29 are canceled to limit the number of claims in the application to 25 to comply with new rules. Claims 1, 2, 4, 11, 12, 14, 17, 21, 22, 24, and 27 are amended. Independent claims 1, 11, and 21 are amended to clarify an "out of process transaction" and to replace the "/" in the preamble with the word "and." Support for the amendments to claims 1, 2, 11, 12, 21, and 22 is located at least on page 10, line 25, through page 11, line 29; on page 16, lines 8-13; and on page 18, lines 6-9, of the specification. Claims 4, 14, and 24 are amended to include the subject matter of canceled claims 5, 15, and 25, respectively. Claims 17 and 27 are amended to include the subject matter of canceled claims 19 and 29, respectively. Reconsideration of the claims is respectfully requested.

I. Interview Summary

Applicants thank Examiner Kimbleann Verdi for the courtesies extended to Applicants' representatives during the November 20, 2007 telephone interview. During the interview, Applicants' representatives discussed the distinctions between the present inventions and the *Avakian* reference. Proposed claim amendments were also discussed. The Examiner indicated that a further search would be conducted to evaluate the clarified features of the amended claims. The substance of the telephone interview is included in the following remarks.

II. 35 U.S.C. § 101

The Examiner has rejected claims 21-30 under 35 U.S.C. § 101 as being directed towards non-statutory subject matter. This rejection is respectfully traversed.

The Office Action states:

> With respect to claims 21-30, the "computer readable medium," in accordance with Applicant's specification, may be a signal bearing media, for example radio frequency and light wave transmissions. This subject matter is not limited to that which falls within a statutory category of invention because it is not limited to a process, machine, manufacture, or a composition of matter. Instead, it includes a form of energy. Energy does not fall within a statutory category since it is clearly not a series of steps or acts to constitute a process, not a mechanical device or combination of mechanical devices to constitute a machine, not a tangible physical article or object which is some form of matter to be a product and constitute a manufacture, and not a composition of two or more substances to constitute a composition of matter.

Office Action dated August 23, 2007, pages 2-3.

Paragraph [0043] of the specification states that the processes of the present invention have been described in the context of a fully functioning data processing system distributed in the form of a computer readable medium of instructions. The computer readable media includes recordable-type media, such as a floppy disk, a hard disk drive, a RAM, CD-ROMs, and DVD-ROMs. Therefore, Applicants respectfully request withdrawal of the rejection of claims 21-30 under 35 U.S.C. § 101.

III. 35 U.S.C. § 102, Anticipation Based on Avakian

The Examiner has rejected claims 1-30 under 35 U.S.C. § 102(e) as being anticipated by *Avakian et al.*, (United States Publication No.: 2005/0039171 A1), hereinafter referred to as *Avakian*. This rejection is respectfully traversed.

With respect to independent claims 1, 11, and 21, the Office Action states:

> 6. As to claim 1, Avakian teaches a method for dynamically monitoring and linking cross-process/cross-thread transactions in a bytecode injected application, the method comprising the computer implemented steps of:
>
> inserting a bytecode inserted probe into the bytecode injected application (paragraphs [0051]-[0052]), wherein the bytecode inserted probe detects a correlating token in an inbound request (paragraphs [0155 and 0177]), retrieves the correlating token and dynamically determines if the inbound request is a child of an out of process transaction (paragraphs [0156] and [0158]);
>
> responsive to a determination that the inbound request is a child of an out of process transaction, recording the inbound request (paragraph [0159]); and
>
> linking the inbound request to the out of process transaction (paragraph [0159]).
>
> ...
>
> 16. As to claims 11-20, these claims are rejected for the same reasons as claims 1-10 respectively, see the rejections to claims 1-10 above.
>
> 17. As to claims 21-30, these claims are rejected for the same reasons as claims 1-10 respectively, see the rejections to claims 1-10 above.

Office Action dated August 23, 2007, pages 3-5.

Claim 1, which is representative of rejected independent claims 11 and 21 with regard to similarly recited subject matter, reads as follows:

> 1. A method for dynamically monitoring and linking cross-process and cross-thread transactions in a bytecode injected application, the method comprising the computer implemented steps of:
>
> inserting a bytecode inserted probe into the bytecode injected application, wherein the bytecode inserted probe detects a correlating token in an inbound request, retrieves the correlating token and dynamically determines if the inbound request is a child of an out of process transaction, wherein the out of process transaction began in a cross-process or cross-thread;

responsive to a determination that the inbound request is a child of an out of process transaction, recording the inbound request; and

linking the inbound request to the out of process transaction.

(emphasis added)

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). Applicants respectfully submit that *Avakian* does not identically show every element of the claimed invention arranged as they are in the claims. Specifically, *Avakian* does not teach or suggest "inserting a bytecode inserted probe into the bytecode injected application, wherein the bytecode inserted probe detects a correlating token in an inbound request, retrieves the correlating token and dynamically determines if the inbound request is a child of an out of process transaction, wherein the out of process transaction began in a cross-process or cross-thread," as recited in independent claims 1, 11, and 21.

Avakian is directed to a method for monitoring performance of a plurality of transactions in a J2EE application server. The transactions include a top level transaction and plurality of transactions relating to the top level transaction in a child parent hierarchy. For each of selected ones of said transactions, the method comprises instrumenting the transaction at run-time without modifying its source code to obtain a performance metric corresponding thereto. Further, for each of said instrumented transactions, a correlator is generated for identifying the top level transaction and a parent, if any, of the transaction. The method further calls for utilizing the correlators to cross-correlate a performance metric corresponding to a parent transaction with one or more performance metrics corresponding to one or more child transactions of said parent transaction. See *Avakian*, Abstract. *Avakian* does not teach or suggest "inserting a bytecode inserted probe into the bytecode injected application, wherein the bytecode inserted probe detects a correlating token in an inbound request, retrieves the correlating token and dynamically determines if the inbound request is a child of an out of process transaction, wherein the out of process transaction began in a cross-process or cross-thread," as recited in independent claims 1, 11, and 21.

With respect to the rejection of claims 1, 11, and 21, the Office Action refers to the following portions of *Avakian*:

> **[0051]** As discussed above, a transaction monitoring system according to the teachings of the invention can include an instrumentation engine that can be utilized for instrumenting selected methods and/or functions associated with software components hosted, for example, on application servers employed in a multi-tier Web architecture. Java-based applications constitute one category of such software components whose selected methods can be instrumented by utilizing methods and systems, and more particularly transaction monitoring agents, provided by the invention, as discussed in more detail below.
>
> **[0052]** In particular, a system of invention can include a bytecode instrumentation engine that can be utilized to modify the bytecode associated with a Java application at any time prior to, or during, the loading and initialization of the bytecode by a Java virtual machine (JVM). More specifically, the bytecode instrumentation engine can include a tool, herein referred to as Bytecode Instrumentation Program (BIP), and another tool, herein referred to as Bytecode Instrumentation Controller (BIC), that can modify methods of classes associated with a Java application prior to being loaded by a JVM or as they are loaded by a JVM.

Avakian, paragraphs [0051-0052].

These portions of *Avakian* disclose a bytecode instrumentation engine that can modify methods of classes associated with a Java application prior to being loaded by a JVM. This is not the same as inserting a bytecode inserted probe into the bytecode injected application. The bytecode inserted probe of the present invention is a component monitor. The bytecode inserted probe detects and retrieves a correlating token in an inbound request and then determines if the inbound request is a child of an out of process transaction that began in a cross-process or cross-thread. *Avakian* does not teach or suggest "inserting a bytecode inserted probe into the bytecode injected application, wherein the bytecode inserted probe detects a correlating token in an inbound request, retrieves the correlating token and dynamically determines if the inbound request is a child of an out of process transaction, wherein the out of process transaction began in a cross-process or cross-thread," as recited in independent claims 1, 11, and 21.

In addition, the Office Action refers to the following portions of *Avakian* with respect to the rejection of claims 1, 11, and 21:

> **[0155]** Because Transaction A is instrumented in accordance with the teachings of the invention, it includes calls to the ARM agent **48 (FIGS. 15 and 16)**. In particular, the ARM agent **48** can be invoked by the ExecCallback interface **36 (FIGS. 2 and 3)** at the start of the instrumented transaction A, as described above. Upon invocation, the ExecCallback interface **36** will access a JTL **64** associated with the logical thread **60** in order to obtain context information related to the Transaction A, and any other control information that may be

needed to influence the behavior of subsequently invoked transactions, e.g., methods or software components. However, since Transaction A is the top level transaction in this exemplary transaction chain, the JTL **64** does not initially include such information. Accordingly, in preferred embodiments of the invention, the ExecCallback interface **36** will generate a correlator associated with Transaction A, and will store this correlator, herein referred to as correlator A, in one location of a stack associated with the JTL **64**.

[**0156**] The correlator A, and subsequently generated correlators associated with Transactions B and C described below, can include a number of fields, each of which provides selected context information regarding the respective transaction. With reference to **FIG. 18**, in this exemplary embodiment, a correlator **66**, e.g., the correlator A, includes a field **66a** that identifies the top level transaction in a hierarchical transaction chain, and a second field **66b** that identifies the parent of a current transaction.

Avakian, paragraphs [0155-0156].

[**0158**] For the top level correlator, the Top and Parent fields provide the same context information, namely, information related to the top level transaction. Alternatively, in this correlator, the parent transaction field can be a null field.

[**0159**] Referring again to **FIG. 17**, the exemplary Transaction A can invoke a child Transaction B that is also instrumented in accordance with the teachings of the invention. The correlator A stored in the JTL **64** will be accessed at an instrumentation point associated with Transaction B, and further, a new correlator, herein referred to as correlator B, will be generated to identify Transaction B. The exemplary correlator B, shown schematically as correlator 68 in **FIG. 18**, is stored in the stack associated with the JTL **64**, and identifies Transaction A as the top level transaction and B as the parent transaction.

Avakian, paragraphs [0158-0159].

[**0177**] In other embodiments, Java Message Service (JMS), which support both point-to-point and publish/subscribe messaging models, can be employed to pass correlators between different processes. For example, a correlator can be incorporated in a properties header of a message that is transmitted from one instrumented transaction executing in one process to another instrumented transaction executing in a separate process.

Avakian, paragraph [0177].

These portions of *Avakian* teach that when a transaction is instrumented, it includes calls to the ARM agent to generate a correlator associated with Transaction A. *Avakian* teaches that the transactions and associated correlators of a single thread are stored in a stack structure associated with a Java Thread Local storage (JTL). *Avakian* does not teach or suggest "inserting a bytecode inserted probe into the bytecode injected application, wherein the bytecode inserted probe detects a correlating token in an inbound request, retrieves the correlating token and dynamically determines if the inbound request is a child of an out of process transaction, wherein

the out of process transaction began in a cross-process or cross-thread," as recited in independent claims 1, 11, and 21.

In addition, *Avakian* states that its transaction monitoring system can convey context information, ie. correlators, from one Java method invocation to another related Java method within a single thread of execution by storing information in the JTL, as shown in paragraphs [0152-0153] below:

> **[0152]** A description of an implementation according to the teachings of the invention for passing correlators from one J2EE transaction to another will follow, and a corresponding discussion relating to COM objects will be provided further below. The J2EE standard defines a request context available to each Web facing component. It, however, does not provide any mechanism for propagating application context between Java methods other than by parameter passing or global variables. A transaction monitoring system of the invention, however, provides a mechanism for propagation of such application context among **related** Java methods, as described below.
>
> **[0153]** More particularly, a transaction monitoring system of the invention can convey context information, e.g., correlators, from one Java method invocation to another within **a single thread of execution** by storing this information in Java Thread Local storage (JTL). Moreover, context information corresponding to each level of interest in an application's dynamic chain of execution can be maintained by providing a "last-in-first-out" (LIFO) stack structure in the JTL.

Avakian, paragraphs [0152-0153]. (emphasis added)

To the contrary, claims 1, 11, and 21 recite a method for dynamically monitoring and linking cross-process and cross-thread transactions in a bytecode injected application. As recited in *Avakian's* claim 15, *Avakian* also teaches a method for monitoring at least two transactions executing in two separate process and being related as parent-child transaction. A correlator is generated for the parent or top-level transaction. This correlator is sent to the child transaction, and then a correlator is generated for the child transaction. The metrics of each are correlated. This is different than the teachings of the present invention, where a probe dynamically determines if the inbound request is a child of a cross-thread or cross-process transaction by detecting and retrieving a correlating token of the inbound request.

In view of the above, Applicants respectfully submit that *Avakian* does not teach each and every feature of independent claims 1, 11, and 21, as is required under 35 U.S.C § 102(e). In addition, *Avakian* does not teach each and every feature of dependent claims 2-4, 6-10, 12-14, 16-18, 20, 22-24, 26-28, and 30 at least by virtue of their dependency on claims 1, 11, and 21, respectively. Claims 5, 15, 19, 25, and 29 are canceled. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 1-30 under 35 U.S.C § 102(e).

In addition to being dependent on their respective independent claims, claims 2-4, 6-10, 12-14, 16-20, 22-24, and 26-30 are also distinguished over the *Avakian* reference based on the specific features recited therein. With respect to amended claims 2, 12, and 22, *Avakian* does not teach or suggest that "the bytecode inserted probe uses any of a plurality of methods to detect and retrieve the correlating token from the inbound request, and wherein the correlating token is passed in the inbound request across thread boundaries and process boundaries using the plurality of methods including at least attaching the correlating token to one of an HTTP request, an outbound JMS message, a Common Object Request Broker Architecture (CORBA) message, and a Simple Object Access Protocol (SOAP) header of a web service request." Specifically, *Avakian* does not teach or suggest that the bytecode inserted probe uses any of a plurality of methods to detect and retrieve the correlating token from the inbound request.

In addition, *Avakian* does not teach or suggest that "the bytecode inserted probe detects the correlating token in the inbound request using a TransactionInfo object," as recited in claims 3, 13, and 23. *Avakian* does not teach or suggest a bytecode inserted probe as recited in the claims or that the bytecode inserted probe detects a correlating token. Further, *Avakian* does not mention using a TransactionInfo object to detect a correlating token.

With respect to claims 6, 16, and 26, *Avakian* does not teach or suggest steps, means, or instructions for linking the inbound request to the out of process transaction is performed by a transaction performance monitor. To the contrary as discussed above, *Avakian* discloses that its transaction monitoring system conveys context information, e.g., correlators, from one Java method invocation to another related Java method within a single thread of execution by storing this information in Java Thread Local storage.

IV. Conclusion

It is respectfully urged that the subject application is patentable over *Avakian* and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: November 21, 2007

Respectfully submitted,

/Gerald H. Glanzman/

Gerald H. Glanzman
Reg. No. 25,035
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants

GHG/VJA